Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 171 (2007) 586–605

Artificial Intelligence

www.elsevier.com/locate/artint

Learning, detection and representation of multi-agent events in videos

Asaad Hakeem *, Mubarak Shah

Computer Vision Lab, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA

Received 5 July 2006; received in revised form 2 April 2007; accepted 9 April 2007

Available online 14 April 2007

Abstract

In this paper, we model multi-agent events in terms of a temporally varying sequence of sub-events, and propose a novel approach for learning, detecting and representing events in videos. The proposed approach has three main steps. First, in order to learn the event structure from training videos, we automatically encode the *sub-event dependency graph*, which is the learnt event model that depicts the conditional dependency between sub-events. Second, we pose the problem of event detection in novel videos as clustering the maximally correlated sub-events using normalized cuts. The principal assumption made in this work is that the events are composed of a highly correlated chain of sub-events that have high weights (association) within the cluster and relatively low weights (disassociation) between the clusters. The event detection does not require prior knowledge of the number of agents involved in an event and does not make any assumptions about the length of an event. Third, we recognize the fact that any abstract event model should extend to representations related to human understanding of events. Therefore, we propose an extension of CASE representation of natural languages that allows a plausible means of interface between users and the computer. We show results of learning, detection, and representation of events for videos in the meeting, surveillance, and railroad monitoring domains.

Keywords: Event learning; Event detection; Temporal logic; Edge weighted directed hypergraph; Normalized cut; Event representation; P-CASE

1. Introduction

An instance of an event is a composition of directly measurable low-level movements, which we term sub-events, having a temporal order. For example, a voting event is composed of a sequence of move, raise, and lower hand sub-events. Also, the agents may act independently (e.g. voting) as well as collectively (e.g. touch-down in a football game) to perform certain events. Hence, in the enterprise of machine vision and artificial intelligence, the ability to detect and represent the observed events is one of the ultimate goals. Also, the practical need for formal representation of events is best illustrated through these applications:

(1) *Surveillance*: By definition, surveillance applications require the detection of peculiar events. Event representations are used for prior definition of what constitutes an interesting event in any given domain, allowing automation of area surveillance.

* Corresponding author.
E-mail addresses: ahakeem@cs.ucf.edu (A. Hakeem), shah@cs.ucf.edu (M. Shah).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2007.04.002

(2) *Annotation and Indexing*: In the spirit of MPEG-7, video sequences are autonomously annotated based on their content using an event representation of the occurred events.

(3) *Event Browsing*: Given a query for a certain event defined in terms of an event representation, similar instances are retrieved from a database of annotated clips.

Given a set of training videos of an event, such that each video consists of a particular instance of the event, we first learn the event models. This process of constructing event models using training videos is called *event learning*. Since we know the type of event in each training video, it is considered supervised learning. Using the learnt event models, we then find events in novel videos, and this process is termed *event detection*. The primary objective of this work is to detect the complex interactions of multiple agents performing multiple sub events, which constitute an event. We do not assume any prior knowledge about the number of agents involved in the interaction and length of the event. Another objective of this work is to present a coherent representation of these events, as a means to encode the relationships between the agents and objects participating in an event.

In order to learn the events from training videos, we detect the sub-events performed by different agents. The complete temporal order of sub-events occurring in a video are encoded in a graph, which is further utilized in extracting the conditional dependency between sub-events. Thus, we determine the *event dependency graph*, which is the learnt event model that encodes the higher order Markov dependencies between sub-events and is scalable to the number of agents involved in the event. Event detection in novel videos proceeds by estimating a weight matrix of conditional dependencies between the detected sub-events. The weights on edges between sub-events are recovered using the learnt event model. This weight matrix is then used for spectral graph partitioning. Thus, *normalized cut* is applied recursively to this weight matrix, to cluster the highly correlated sub-events. These clusters represent the detected events for a specific event model, and the event structure composed of sub-events and their temporal order is extracted. This process of estimating weight matrix and clustering of sub events is repeated for each event model.

Finally, in order to represent the events, we extend the CASE [12] representation of the natural language. There are several other event representations (see next section for details) proposed in the literature such as hierarchical Event Representation Language (ERL) [31], graph representation of object trajectories [29], Bayesian Networks based event representation [20], and video event ontologies [32] that provide varying degrees of representation to the actions and agents involved in the events. However, these works did not address important issues of temporal and causal relationships. Moreover, no mechanisms were proposed for multiple-agents or multi-threaded events.

CASE was primarily used for syntactic analysis of natural languages. While it provides a promising foundation for event representation, it has several limitations for that end. We therefore propose two critical extensions to CASE for the representation of events: (1) accommodating multiple agents and multiple threads in an event, and (2) supporting the inclusion of temporal information into the representation. Further, we propose a novel event graph representation for the detected events in video sequences that have temporal relationships between sub-events. Hence, unlike almost all the previous work, we use both temporal structure and an environment descriptor simultaneously to represent an event. We also recognize the importance of representing the variations in temporal order of sub-events that occur in an event and encode it directly into our representation, which we term P-CASE. These variations in the temporal order of sub-events occur due to the style of execution of events for different agents. We empirically demonstrate our framework for event detection and representation in the meeting, surveillance, and railroad monitoring domains.

The rest of the paper is organized as follows. In the next section, we present the related work and their limitations for multiple agent event detection and representation. The event learning process is summarized in Section 3, where we present the formulation of the probabilistic event model. While the event detection using the learnt event model is detailed in Section 4. The representation of the detected events is presented in Section 5, where the CASE representation is extended to allow a plausible means of event representation. Finally, the results and discussion of our experiments are demonstrated in Section 6 followed by the conclusion of our work in Section 7.

2. Related work

In the literature, a variety of approaches have been proposed for event detection in video sequences. Most of these approaches can be organized into three main categories. First, there are approaches that detect events in videos based on pre-defined event models in the form of templates, rules, or constraints. These models are typically hand crafted.

Among these include methods that utilize motion verbs [25], intermodal collaboration using domain knowledge [4], spatio-temporal motion segmentation [40], indexing of pose and velocity vectors of body parts [7], image ontology based recognition [27], and taxonomy and ontology of domains [16]. Second, approaches that automatically learn the event models [9,13,19,22] using training data, which have been widely used in the area of event detection. Third, approaches that do not explicitly model the events [39,52,53], but utilize clustering methods for event detection.

Initial approaches for event detection involved methods that had pre-defined rules or heuristic constraints which formed event models. Among these methods were approaches that modeled events using state machines. Badler [5] proposed event models using state graphs and primitive rules on artificial environments. The method used prior knowledge of the environment to resolve complex events rather than motion information for event detection. Ayers and Shah [3] also used state machines for event detection, however as opposed to the above method, they utilized the motion data for event detection. Their method detected events in specified areas, which restricted the system and required a priori knowledge of the environment. Haering et al. [15] proposed a three level event detection model, which applied neural networks on low-level features for object classification and shot summarization, and state machines as the high level event model that detected events based on the output of the low and mid-level models. Unlike previous methods, their method did not require prior knowledge of the environment for event detection. Further approaches utilized constraints and grammar rules for the detection of events that had variability and did not strictly follow the event model. The methods in this category employ a variety of techniques including causal grammar [8], dynamic perception of actions [28], angular constraints on body components [1], stochastic grammars [30], and appearance and geometric constraints [46]. All the above approaches either manually encode the event models or provide constraints such as grammar or rules to detect events in novel videos.

Next, we discuss approaches that automatically learn the event models using training data instead of manually encoding the event models. Friedman et al. [13] detected events by learning the structure and parameters of Dynamic Bayesian Networks (DBNs) from both complete and incomplete data. Though their method was partially insensitive to occlusion, it was limited to inference of simple events, and required approximations for detecting complex events. Methods that adopt Hidden Markov Models (HMMs) [9,35] and its variations such as coupled HMMs [34], dynamic multi-linked HMMs [14], abstract hidden Markov memory models [33], and layered HMMs [51] have been widely used in the area of event detection. These methods were the work horses for event detection in the surveillance, indoor and office environments, and sports domains. Ivanov and Bobick [22] proposed a Stochastic Context Free Grammar (SCFG) for event detection. The grammar rules were assigned probabilities based on the training data and were further utilized for event detection in novel videos. The primitive events were first detected using HMMs and subsequently parsed by the SCFG for error detection and correction of the detected events. Other methods for event detection using learnt event models include PNF (past-now-future) propagation of temporal constraints [38], belief networks [21], force dynamics [45], shape activities [47,48], Support Vector Machines (SVMs) [19], Bayesian networks and probabilistic finite state machines [20], propagation networks of partially ordered sequential action [43] and partially labeled data [44], mutual invariants of 3D joint motion [36,37], and HMMs and multi-class Adaboost on 3D joint motion [26]. The above learning methods either model single person events or require prior knowledge about the number of people involved in the events and variation in data may require complete re-training, so as to modify the model structure and parameters to accommodate those variations. Also, there is no straight-forward method of expanding the domain to other events, once training has been completed. That is, if more events are added to the current domain or if we want to model events in a new domain, then, the existing models are re-trained using the new data and/or the model structure is defined manually for the new events.

Finally, we describe approaches that do not explicitly model events but utilize clustering techniques for event detection. These methods of event detection include spatio-temporal derivatives [52] and co-embedding prototypes [53]. Both methods find event segments by spectral graph partitioning of the weight matrix. The weight matrix is estimated by calculating a heuristic measure of similarity between video segments. These methods assumed maximum length of an event and were restricted to a single person and a single threaded event detection. Rao et al. [39] proposed human action recognition using spatio-temporal curvatures of 2D trajectories. Their method initiated without an event model and formed clusters of similar events based on their spatio-temporal curvatures. Their method is also restricted to single person event detection, but it makes no assumptions about the length of an event and the spatio-temporal curvature based event representation was also view-invariant.

What is missing in these approaches is the ability to model long complex events involving multiple agents performing multiple actions. Can these approaches be used to automatically learn the events involving an unknown number

of agents? Will the learnt event model still hold for a novel video in case of interfering events from an independent agent? Can these approaches extend their abstract event model to representations related to human understanding of events? Can events be compared on the basis of these representations? These questions are addressed in this paper. In our approach, event models are learnt from training data, and are used for event detection in novel videos. *Event learning* is formulated as modeling conditional dependencies between sub-events, while *event detection* is treated as a graph-theoretic clustering problem. The novelty of our method compared to the above mentioned methods is the ability to detect the multiple agents performing multiple actions in the form of events, without prior knowledge about the number of agents involved in the interaction and length of the event. Further, we present a coherent representation of these events, as a means to encode the relationships between agents and objects participating in an event.

The above mentioned questions are discussed in detail in their respective sections but are briefly answered here for completeness. Almost all event models require prior knowledge about the number of agents involved in the event, where as our learnt event model is scalable to the number of agents involved in an event. The reason is that instead of modeling agent processes (like Hidden Markov Models, Dynamic Bayesian Networks etc.) we model the sub-event dependency for all agents simultaneously i.e. which sub-event occurs more frequently after another sub-event for a given event. Thus, our event model is agnostic to the number of agents involved in the event. Also, since most methods detect events by estimating a posterior probability of a sub-event sequence, independent agent actions in the sub-event sequence reduces the overall posterior probability, where as we use graph clustering techniques for event detection. Thus, our method clusters events with high edge weights within the cluster and segments out independent agent actions as those actions have low edge weights with the rest of the sub-events belonging to the event. Furthermore, most event models and representations are abstract and complex, with no notion of human readability, where as our event representation is an extension of the CASE representation that was used for syntactic analysis of the natural language. It has explicit cases for agents, location, dative etc. that completely describes the event. We believe it is easier for humans to relate to this event representation.

3. Learning the event structure

In this section, we describe how to learn the event structure from training videos. Let $f(\mathbf{p}, t)$ represent a continuous video signal, indexed by spatial and temporal coordinates respectively. By indexing on the discrete-time variable k, we *temporally* represent the video signal as the set $\{f[\mathbf{x}, k]\}$ for $1 \leqslant k \leqslant N$, where N is the temporal support of the video signal, and $\mathbf{x} = (x, y)$ denotes the spatial coordinate. Each entity is represented in terms of its label (person, object, hand, or head) and motion, e.g. $\{\mathsf{vehicle}_a, \mathbf{u}_a\}$, where $\mathbf{u}_a = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is the trajectory of $\mathsf{vehicle}_a$'s centroid. Here, it is assumed that the lower-level tasks of object detection, classification and tracking have been performed. Also, it is important to note that since it is the *relative* concept of motion that we are interested in (e.g. where did agent_1 move to with respect to object_2?), two-dimensional projections of three-dimensional world trajectories are sufficient for event representation (barring some degenerate configurations). An example video of the stealing event is shown in Fig. 1. These sub-events are input into a system that represents them in terms of a sub-event dependency graph which is described next.

3.1. Sub-event dependency graph

The detected sub-events are represented in a sub-event dependency graph (SDG). A naive formulation of the problem would be to consider a complete graph for estimating the conditional dependencies between sub-events.

Fig. 1. Automated detection of sub-events for stealing video. Using the tracked trajectories, the sub-events of each agent are detected, and frames 37, 127, and 138 of the video are shown.

The problem with the complete graph formulation is that the sub-events are not dependent on *all* their predecessor sub-events, rather they are dependent on their proximal predecessor sub-events. For example, a person *raising* a hand sub event at the start of the video is not related to *picking* a book sub-event, occurring after a few minutes. Thus, *transitive reduction* based upon proximity x is applied to the graph. This does not imply that we constrain our events to be a maximum of x length, instead it denotes that the events are composed of $(x-1)$th order Markov chain of sub-events. That is, each sub-event is conditionally dependent on (at most) $x-1$ parent sub-events, which is true for most of the events in the considered domains.

An SDG is a hypergraph $G = (V, E, W)$ having a number of *vertices* $V = \{v_1, v_2, \ldots, v_n\}$ which represents n *unique* sub-events present in an event. *Hyperarcs* $E = \{e_1, e_2, \ldots, e_m\}$ are backward arcs (*B-arcs*), where each B-arc is an *ordered* pair of vertices $e_i = (P_i, v_i)$ such that $P_i \subseteq V$, and P_i represents the temporally ordered parent sub-events of v_i. Additionally, $W = \{w_1, w_2, \ldots, w_m\}$ are the *weights* on the B-arcs, which represent the conditional dependencies of child sub-events upon a sequence of parent sub-events.

An ordinary graph is a 2-uniform hypergraph, where k-uniform signifies that each hyperedge has a *cardinality* of k vertices. We do not enforce a k-uniform hypergraph, instead we allow the hypergraph to have a maximum x edge cardinality (4 in our experiments). This allows the encoding of conditional probabilities of sub-events v_i with a maximum of $x-1$ parent sub-events. The equations for estimating the weights w_i on hyperarcs e_i for cardinality of $X \in \{2, 3, 4\}$ are respectively given by (1), (2), and (3):

$$P\left(v_i^t | v_j^{t-1}\right) = \frac{P(v_i^t, v_j^{t-1})}{P(v_j^{t-1})}, \tag{1}$$

$$P\left(v_i^t | v_j^{t-1}, v_k^{t-2}\right) = \frac{P(v_i^t, v_j^{t-1}, v_k^{t-2})}{P(v_j^{t-1} | v_k^{t-2}) P(v_k^{t-2})}, \tag{2}$$

$$P\left(v_i^t | v_j^{t-1}, v_k^{t-2}, v_l^{t-3}\right) = \frac{P(v_i^t, v_j^{t-1}, v_k^{t-2}, v_l^{t-3})}{P(v_j^{t-1} | v_k^{t-2}, v_l^{t-3}) P(v_k^{t-2}, v_l^{t-3})}, \tag{3}$$

where v_i^t represents a sub-event i occurring at (time) index t, and $Agent(v_i^t) = Agent(v_a^b)$ $(a \in \{j, k, l\}, b \in \{t-1, t-2, t-3\})$, which enforces that the current and parent sub-events are to be performed by the *same* agent. This is necessary since the sub-events performed by independent agents are not conditionally dependent on each other. If both the agents are involved in a sub-event, there is a conditional dependency between their sub-events. Eq. (1) represents the conditional probability of a sub-event v_i occurring at time index t, given that sub-event v_j occurred at $t-1$. Similarly, equation (2) represents the conditional probability of sub-event v_i occurring at t, given that sub-event v_j occurred at $t-1$, which was preceded by the sub-event v_k that occurred at $t-2$. An example of a partial SDG estimated from a video containing *voting* events is given in Fig. 2. Note that the SDG captures all the variations in temporal order of sub-events as well as the conditional dependencies of all sub-events in a video. The SDG is also scalable to the number of agents involved in an event since it accumulates the conditional dependency between sub-events that are being performed by various agents.

4. Event detection in novel videos

Given learnt event models ξ_i in a supervised manner, *event detection* deals with automatically detection of those events in novel videos. In our approach, first a graph of detected sub-events in a novel video is constructed, and a weight matrix of conditional dependencies between sub-events using the learnt event model is computed. This weight matrix is used for spectral graph partitioning. Thus, *normalized cut* is applied recursively to this weight matrix in order to cluster the highly correlated sub-events. These clusters represent the detected events for a specific event model and the event structure composed of sub-events and their temporal order is extracted. This process of estimating weight matrix and clustering is repeated for each event model.

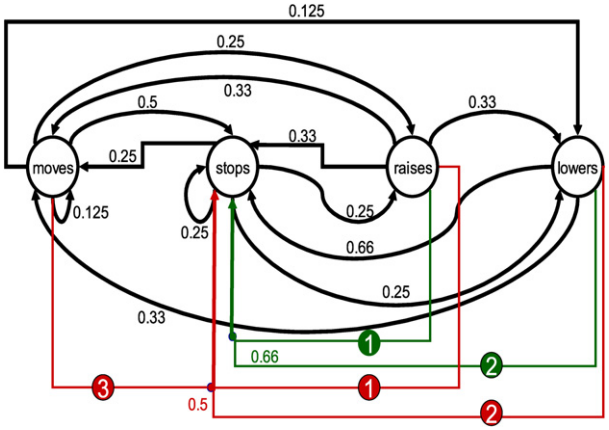


Fig. 2. Partial sub-event dependency graph for a sample video containing multiple people performing *voting* events in various styles. The vertices represent the sub-events, while the conditional probabilities between sub-events are represented by the weights on the hyperarcs. Note that a single example of hyperarcs with cardinality of 3 and 4 are shown respectively in green and red for clarity. The circled number on the hyperarc represents the temporal order index in P_i e.g. the B-arc of cardinality 4 represents P(stops | moves, lowers, raises) i.e. the probability of 'stop' occurring after a sequence of 'raises', 'lowers', and 'moves' sub-events. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Estimating the probabilistic weight matrix

In order to determine the probabilistic weight matrix for a specific event model ξ_p, we generate a graph G of the detected sub-events in the novel video and obtain θ_p (edge weights representing conditional probabilities) from the learnt event model (SDG), p. Each edge weight w_{ij} between two nodes of G is estimated using:

$$w_{\alpha\beta} = P\left(v_l^t | Pa\left(v_l^t\right)\right) = P\left(v_l^t | v_k^{t-1}, v_j^{t-2}, v_i^{t-3}\right),$$

where α is the index of sub-event v_l^t, and β is the index of (parent of v_l^t) $Pa(v_l^t)$ sub-event. $Pa(v_l^t)$ is the oldest parent sub-event that v_l^t conditionally depends upon such that $Pa(v_l^t) \in \{v_k^{t-1}, v_j^{t-2}, v_i^{t-3}\}$. Note that a sub-event may be dependent upon one or two parent sub-events, hence the conditional probabilities from hyperarcs of cardinality one and two respectively are inserted from the SDG to the weight matrix. In summary, the above weight estimation assigns higher weights to the longer chain of sub-events that occur frequently in the training video of ξ_p. The final weight matrix $\hat{W}_p$ is an upper triangle, since G is a directed acyclic graph. The weight matrix is made symmetric by $\tilde{W}_p = \hat{W}_p + \hat{W}_p^T$ [11], where $\hat{W}_p^T$ is the transpose matrix of $\hat{W}_p$. The *Ncut* minimization function for weight matrices W_p and $\tilde{W}_p$ are equivalent, and the proof is given in Appendix A.

4.2. Event detection using normalized cut

Normalized cut [42] is an unbiased method of partitioning a graph V into two (or more) segments A and B, since it uses a global criterion for graph segmentation rather than focusing on the local features. The global criterion is given by:

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)},$$

where $cut(A, B) = \sum_{u\in A, v\in B} w(u, v)$, $w(u, v)$ is the edge weight between vertices u and v, and $asso(A, V) = \sum_{u\in A, v\in V} w(u, v)$. If the *Ncut* criterion is minimized, the graph is partitioned at the edges with a minimum cut weight. The two partitions have maximum association within and minimum disassociation between their respective partitions. The minimization of the *Ncut* criterion is achieved by finding the second smallest eigenvector of the generalized eigensystem:

$$(D - W)x = \lambda Dx,$$

where, D is a $N \times N$ diagonal matrix with $d(i) = \sum_j w(i, j)$ as the diagonal elements, W is a $N \times N$ symmetric weight matrix, λ and x are the eigenvalues and eigenvectors respectively. The sub-event clustering algorithm using normalized cuts is summarized below:

(1) Compute the weight matrix W, and estimate the diagonal matrix D.
(2) Solve $(D - W)x = \lambda Dx$ to obtain the eigenvector with the second smallest eigenvalue, and use it to bipartition the graph by finding the splitting point such that the *Ncut* is minimized.
(3) Decide if the current partition should be further subdivided by checking that *Ncut* and *average edge weight* (that determines the association within a partition) are below their respective thresholds, and then recursively repartition the segmented parts (if necessary).

The sub-event clusters determined by normalized cuts are the maximally correlated sub-events, given the likelihood estimates of the chain of sub-events. These segmented events have high weights between sub-events within the cluster and relatively low weights between sub-events outside their cluster. An example *weight matrix* estimated using the SDGs of voting and object passing events, and their segmentation obtained after recursive application of *Ncut* is shown in Fig. 3.

5. Event representation

The event clusters obtained through *Ncut*, in the form of temporally related sub-events, may not be that easily interpretable by humans. Although the event clusters represent the order of sub-events performed by various agents in an event, they do not provide details about the event. For example, the object passing event is represented as a temporally related sequence of moves, holds, passes, holds, and moves sub-events. This representation does not provide the details about what was passed, or which hand was used to hold the object. In this section, we extend CASE representation proposed by Fillmore [12] for natural language understanding. The basic unit of this representation is a case-frame, which has several elementary cases, e.g. agentive, instrumental and predicate. However, CASE was primarily used for syntactic analysis of natural languages, and while it provides a promising foundation for event representation, it has several limitations for that end. We therefore propose an extended event representation that addresses the shortcomings of CASE. In particular, we recognize the importance of representing the variations in the temporal order of sub-events occurring in an event, and encode it directly into our representation, which we term P-CASE. These variations in the temporal order of sub-events occur due to the style of execution of events by different agents. Also, we automatically learn the event structure from training videos, and encode the *event ontology* using P-CASE. This has a significant advantage since the domain experts need not go through the tedious task of determining the structure of events by browsing all the videos in the domain.

5.1. CASE framework

The basic unit of this representation is a case-frame, which has several elementary cases. The description of the basic cases used by Fillmore is as follows, explained through an example:

Jack carefully advised Jim about the presentation in the meeting room using examples

(1) **PRED**: predicate
A function name or a label given to the case-frame i.e. *advised*
(2) **AG**: agentive
Main person or entity actively involved in a sub-event i.e. *Jack*
(3) **I**: instrumental
Device used by the agentive during the sub-event i.e. *examples*
(4) **D**: dative
The person or entity affected by the actions of the agentive i.e. *Jim*
(5) **LOC**: locative
The location of the sub-event i.e. *meeting room*

	moves ag1	stops ag1	raises ag1	lowers ag1	raises ag2	moves ag2	lowers ag2	stops ag2	moves ag1	holds ag1	passes ag1,ag2	holds ag2	moves ag2
moves ag1	0	0.1826	0.1027	0.1598	0	0	0	0	0	0	0	0	0
stops ag1	0.1826	0	0.1141	0.0776	0	0	0	0	0	0	0	0	0
raises ag1	0.1027	0.1141	0	0.0717	0	0	0	0	0	0	0	0	0
lowers ag1	0.1598	0.0776	0.0717	0	0.013	0.1928	0	0.2029	0	0	0	0	0
raises ag2	0	0	0	0.013	0	0.877	0.437	0.2087	0.03	0	0.0361	0	0
moves ag2	0	0	0	0.1928	0.877	0	0.4337	0.1826	0.1027	0.1153	0.1923	0	0
lowers ag2	0	0	0	0	0.437	0.4337	0	0.2029	0.1014	0.0203	0.0203	0.0838	0
stops ag2	0	0	0	0.2029	0.2087	0.1826	0.2029	0	0.1141	0.0228	0.0228	0.0228	0.0833
moves ag1	0	0	0	0	0.03	0.1027	0.1014	0.1141	0	0.0457	0.0307	0.0755	0.0211
holds ag1	0	0	0	0	0	0.1153	0.0203	0.0228	0.0457	0	0.1904	0.1842	0.0263
passes ag1,ag2	0	0	0	0	0.0361	0.1923	0.0203	0.0228	0.0307	0.1904	0	0.221	0.0856
holds ag2	0	0	0	0	0	0	0.0838	0.0228	0.0755	0.1842	0.221	0	0.331
moves ag2	0	0	0	0	0	0	0	0.0833	0.0211	0.0263	0.0856	0.331	0

(a)

	moves ag1	stops ag1	raises ag1	lowers ag1	raises ag2	moves ag2	lowers ag2	stops ag2	moves ag1	holds ag1	passes ag1,ag2	holds ag2	moves ag2
moves ag1	0	0.0403	0.0092	0.0015	0	0	0	0	0	0	0	0	0
stops ag1	0.0403	0	0.044	0.0375	0	0	0	0	0	0	0	0	0
raises ag1	0.0092	0.044	0	0.0174	0.017	0.15	0.12	0	0	0	0	0	0
lowers ag1	0.0015	0.0375	0.0174	0	0.0563	0.002	0.012	0.0005	0	0	0	0	0
raises ag2	0	0	0.017	0.0563	0	0.266	0.132	0.087	0	0	0	0	0
moves ag2	0	0	0.15	0.002	0.266	0	0.274	0.126	0	0	0	0	0
lowers ag2	0	0	0.12	0.012	0.132	0.274	0	0.291	0.045	0.12	0	0	0
stops ag2	0	0	0	0.0005	0.087	0.126	0.291	0	0.2445	0.1322	0.017	0	0
moves ag1	0	0	0	0	0	0	0.045	0.2445	0	0.4766	0.371	0.325	0.299
holds ag1	0	0	0	0	0	0	0.12	0.1322	0.4766	0	0.364	0.285	0.2167
passes ag1,ag2	0	0	0	0	0	0	0	0.017	0.371	0.364	0	0.2201	0.195
holds ag2	0	0	0	0	0	0	0	0	0.325	0.285	0.2201	0	0.1333
moves ag2	0	0	0	0	0	0	0	0	0.299	0.2167	0.195	0.1333	0

(b)

Fig. 3. The estimated *weight matrices* and *Ncut* application for two novel videos. (a) The estimated weight matrix using the SDG of *voting* event. After recursive application of *Normalized cut* algorithm, the two voting events are automatically segmented and are shown as red and blue patches. (b) The estimated weight matrix using the SDG of *object passing* event. After application of *Normalized cut* algorithm, the object passing event is automatically segmented and is shown as the green patch. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(6) **OBJ**: objective
The neutral acted upon by the agentive during the sub-event i.e. *presentation*
(7) **FAC**: factative
The state or action supporting the predicate i.e. *carefully*

The collection of all case-frames forms the CASE framework. The above example in case-frame notation is given below:

[**PRED**: advised, **AG**: Jack, **D**: Jim, **I**: examples, **OBJ**: presentation, **LOC**: meeting room, **FAC**: carefully]

In our representation, the case-frames correspond to sub-events, and the collection of these sub-events form the event. While the CASE framework provides a promising foundation for event representation, it has several limitations to that end, which are discussed in the next section.

5.2. Extended CASE representation

In this section, we discuss the two extensions to the CASE framework. Firstly, in order to capture both multi-agent and multi-thread events, we introduce a hierarchical CASE representation of events in terms of sub-events and case-lists. Secondly, since the temporal structure of events is critical to understanding and hence, representing events, we introduce temporal logic into the CASE representation based on the interval algebra in [2].

Except in constrained domains, events typically involve multiple agents engaged in several dependent or independent actions. Thus any representation of events must be able to capture the composite nature of *real* events. To represent multiple objects, we introduce the idea of case-lists of elements for a particular case. For example, if there are two or more agents involved in an event, CASE cannot represent it in a case-frame as it only allows one agentive in the representation, therefore we add them in a case-list within **AG**,

[**PRED**: move, **AG**:{person1, person2}, ...]

To represent multiple threads, we introduce the concept of sub-event lists (SUB case). An example is shown below to clarify the concept:

"Clyde robs the bank with the help of Bonnie who distracted the cashier by talking to him"

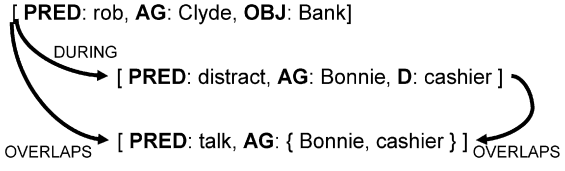
[**PRED**: rob, **AG**: Clyde, **OBJ**: Bank, **SUB**: {
[**PRED**: distract, **AG**: Bonnie, **D**: cashier],
[**PRED**: talk, **AG**:{Bonnie, cashier}]}]

In the above example, there are three sub-events occurring simultaneously, which cannot be represented in CASE representation as it only allows a single sub-event to occur at a particular time. Also, it is evident from the example that the above event representation is ambiguous as temporal relations have not been defined. When did Bonnie talk to the cashier? Was it before, during, or after the rob sub-event? In order to have an unambiguous representation, we need to incorporate temporal relations in our representation.

Events are rarely instantaneous and often largely defined by the temporal order and relationship of their sub-events. In order to represent temporal relationships of sub-events, we introduce temporal logic into the CASE representation based on the interval algebra of [2]. We use this algebra to represent seven temporal relationships[1] as shown in Fig. 4. The entire list of temporal relations for two sub-events T_1 and T_2 is as follows,

AFTER: $T_2^{start} > T_1^{end}$,
MEETS: $T_1^{end} = T_2^{start}$,
DURING: $(T_1^{start} < T_2^{start}) \wedge (T_1^{end} > T_2^{end})$,
FINISHES: $(T_1^{end} = T_2^{end}) \wedge (T_1^{start} < T_2^{start})$,
OVERLAPS: $(T_1^{start} < T_2^{start}) \wedge (T_1^{end} > T_2^{start}) \wedge (T_1^{end} < T_2^{end})$,
EQUAL: $(T_1^{start} = T_2^{start}) \wedge (T_1^{end} = T_2^{end})$,
STARTS: $(T_1^{start} = T_2^{start}) \wedge (T_1^{end} \neq T_2^{end})$.

Since temporal relationship exist between sub-events, they are represented on the directed edges between parent and child sub-events. Consider the above mentioned example of the steal event by Bonnie and Clyde. The case-frames with the temporal logic incorporated are,



[1] A minor modification was made by replacing **BEFORE** with **AFTER** for ease of use.

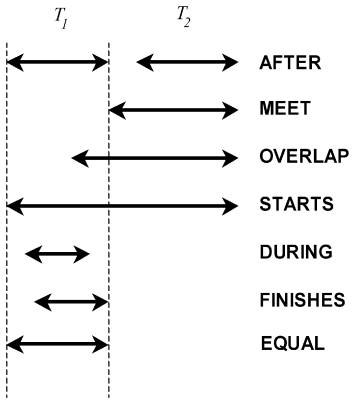
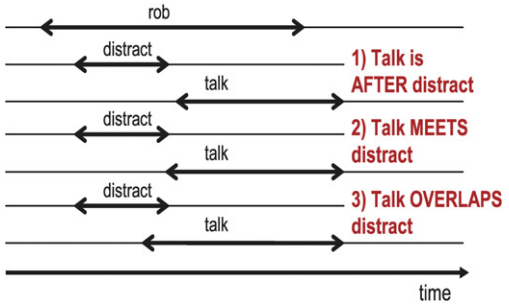


Fig. 4. Allen's interval algebra describing temporal relations between durative sub-events T_1 and T_2.

The above directed event graph representation is an extension of the tree structure of CASE^E [17], where each directed edge represents a temporal parent-child relationship. The tree structure depicted a temporal relation with only its parents and not sibling sub-events (i.e. no OVERLAPS relationship between distract and talk sub-events) and thus had ambiguity in representing the complete order of sub-events. This is explained through the following example:



With sub-event 'distract' occurring DURING 'rob' sub-event, and 'talk' OVERLAPS 'rob' sub-event; there are three temporal possibilities between the 'distract' and 'talk' sub-events: (1) talk is AFTER distract, (2) talk MEETS distract, or (3) talk OVERLAPS distract. Thus, without the temporal relationship between distract and talk, there is ambiguity in temporal order of sub-events in the event representation. This ambiguity in the order of sub-events is resolved using the above graph representation.

5.3. Temporally varying event representation

The above mentioned extensions to CASE provide a plausible interface for event representation. But events rarely occur with the same temporal order of sub-events. The variations in the temporal order of sub-events occur due to the different styles of execution of events by various agents present in the video. Thus, we modify the extended CASE representation to encode the temporal variations present in events.

Given the detected sub-events and SDG for a training video, we estimate the probabilistic weight matrix using the procedure described in Section 4.1. Each vertex in the weight matrix is encoded with a complete case-frame, instead of just the sub-event and agent information. Further application of normalized cuts to the weight matrix segments the different the event instances.

After obtaining the different event instances, the conditional dependencies between sub-events are estimated using Eq. (1) while the variation weights w_i^j in temporal relationships are computed using $w_i^j = \psi(T_i^j)/(\sum_k^n T_k^j)$, where w_i^j denotes the ith weight for the jth edge, $\psi(T_i^j)$ is the frequency of occurrence of the ith temporal relationship for the jth edge, and $\sum_k^n T_k$ is the normalizing factor representing all the n temporal relationships in the interval algebra for the jth edge. The extended CASE representation is further modified by introducing these conditional dependencies

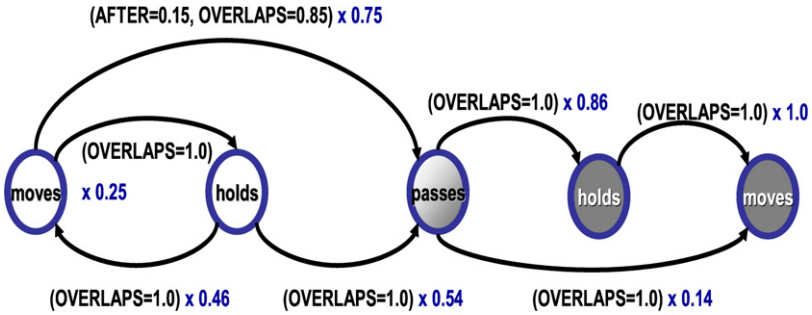


Fig. 5. P-CASE representation for the *object passing* event. Each node is a sub-event encoded by a complete case-frame, and the weights on directed edges represent the probability of occurrence of a specific temporal relationship between sub-events, while the weights outside the brackets (in blue) are the conditional probabilities between sub-events. The white and grey vertices represent sub-events of agents one and two respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and temporal variation weights w_i^j on directed edges of the segmented event graph. The final event representation is termed P-CASE, and an example of object passing event representation is shown in Fig. 5.

One might argue that the SDG should be a sufficient event representation that inherently captures the variations in temporal order of sub-events. It should be noted that the SDG only encodes the conditional dependencies between unique sub-events. In contrast, P-CASE encodes *all* the sub-events with their temporal order that occur in an event. It encodes additional information such as which agent performed what sub-event, that is lost while encoding the SDG. As shown in Fig. 5, P-CASE encodes the variations in moves, holds, passes, holds and moves sub-events, while the SDG would only encode the conditional dependencies between moves, holds, and passes sub-events.

Furthermore, P-CASE has several advantages over existing event representations. First, it simultaneously uses both temporal structure and an environment descriptor to represent an event. Second, various events may have alternate starting sub-events, e.g. 'holds' may be before 'moves' in the given example. The ability to encode events with alternate starting sub-events is yet another advantage that the previous representations lacked. Third, we find the most likely sequence of sub-events by finding the sequence with the maximum likelihood estimate (see details in Appendix B). Finally, the representation is scalable to the number of agents involved in an event. A complete list of the most likely sequence of sub-events extracted from the P-CASE representation of the railroad monitoring domain is provided in Appendix B.

6. Experiments and discussion

We performed experiments for event detection in videos for the meeting, railroad monitoring and surveillance domains. These videos contain multiple agents that act independently or interact with each other or objects. In our experiments, the videos in all domains totaled 194,519 frames comprising of 11,540 sub-events and 1013 events. We

Table 1
Summary of different datasets used for event learning

Dataset name	Videos	Total frames	Events	Sub-events
NIST	6	2480	11	214
Kojima	20	2406	29	264
Sadiye	10	6461	13	104
VACE [56]	40	18724	118	1296
PETS [55]	143	45430	343	2040
CAVIAR [54]	33	28692	91	1507
ETISEO	32	33184	131	1870
Alexei	12	1894	12	48
FDOT	85	14271	98	1524
Meeting	37	4383	37	373
Surveillance	30	15352	35	673
Railroad	46	9595	51	552

Table 2
Detailed description of unique sub-events

Function argument sets:	
Agent = {set of animates, e.g. person, vehicle etc.}	
Object = {set of non-animates, e.g. book, gate, railway signal, etc.}	
Entity = Agent ∪ Object	
Sub-event function definitions:	
Moves(Entity):	function detecting movement of an entity
Stops(Entity):	function detecting seizure of movement of an entity
Enters(Agent):	function detecting entry of an agent in the field of view
Exits(Agent):	function detecting exiting of an agent from the field of view
Approaches(Agent,Entity):	function detecting movement of an agent towards an entity
Leaves(Agent,Entity):	function detecting movement of an agent away from an entity
Extends(Agent,{hand}):	function detecting movement of an agent's hand away from the body
Holds(Agent,Object):	function detecting proximal movement of an agent with an object
Picks(Agent,Object):	function detecting initial movement of an object with an agent
Passes(Agent,Agent,Object):	function detecting movement of object from one agent to another
Drops(Agent,Object):	function detecting seizure of object movement with agent movement
Raises(Agent,{hand}):	function detecting positioning of an agent's hand above head
Lowers(Agent,{hand}):	function detecting positioning of an agent's hand below head
Sits(Agent):	function detecting seizure of movement of an agent with downward motion
Stands(Agent):	function detecting initial movement of an agent with upward motion
Pushes(Agent,Agent):	function detecting quick/short movement of one agent away from the other
Blocks(Agent,Agent,Object):	function detecting occlusion of agent's view of an object by another agent
Crouches(Agent):	function detecting downward movement of an object
Hides(Agent,Object):	function detecting occlusion of an agent with an object
Emerges(Agent,Object):	function detecting reappearance of an agent from behind an object
Collides(Agent,Entity):	function detecting fast movement of an agent into an entity
Breaks(Agent,Entity):	function detecting collision of an agent with change in entity shape
Switches(Object):	function detecting change in object state i.e. signal switching on or off

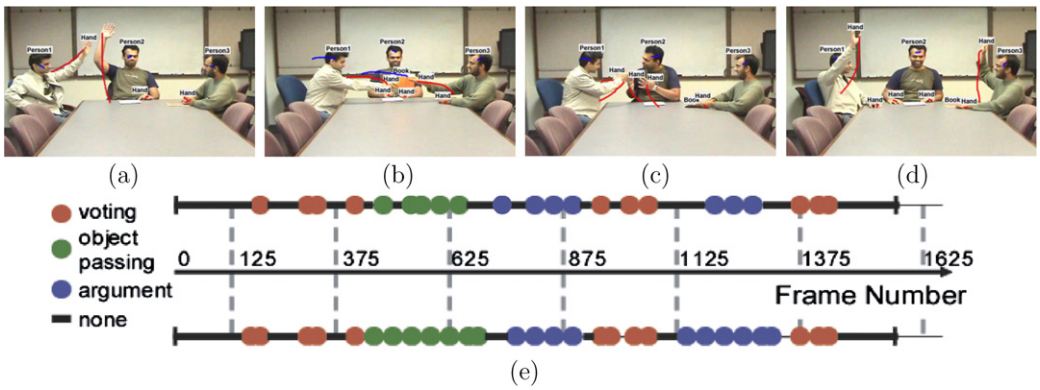


Fig. 6. Event detection results using normalized cuts for meeting domain test video. (a)–(d) represent frames 328, 560, 755, and 1375 respectively of meeting video consisting of 1551 frames. (e) Time indexed clustering results for meeting video where the top bar shows the actual event detection results, and the bottom bar denotes the ground truth of the events.

used seven standard video datasets as well as other videos for training and testing the event detection framework. A total number of 494 videos were adopted for training 16 events. The summary of event learning using different datasets is provided in Table 1. The two sections in the table show the standard dataset and our dataset description respectively.

Initial object identification and labeling were performed manually, and further tracking was attained using MEANSHIFT [10] algorithm. Though techniques like [6,23,24,41,49,50] could have been used for automated object labeling and multiple agent tracking, we opted for a simpler solution as our focus is in event detection. Using the tracked tra-

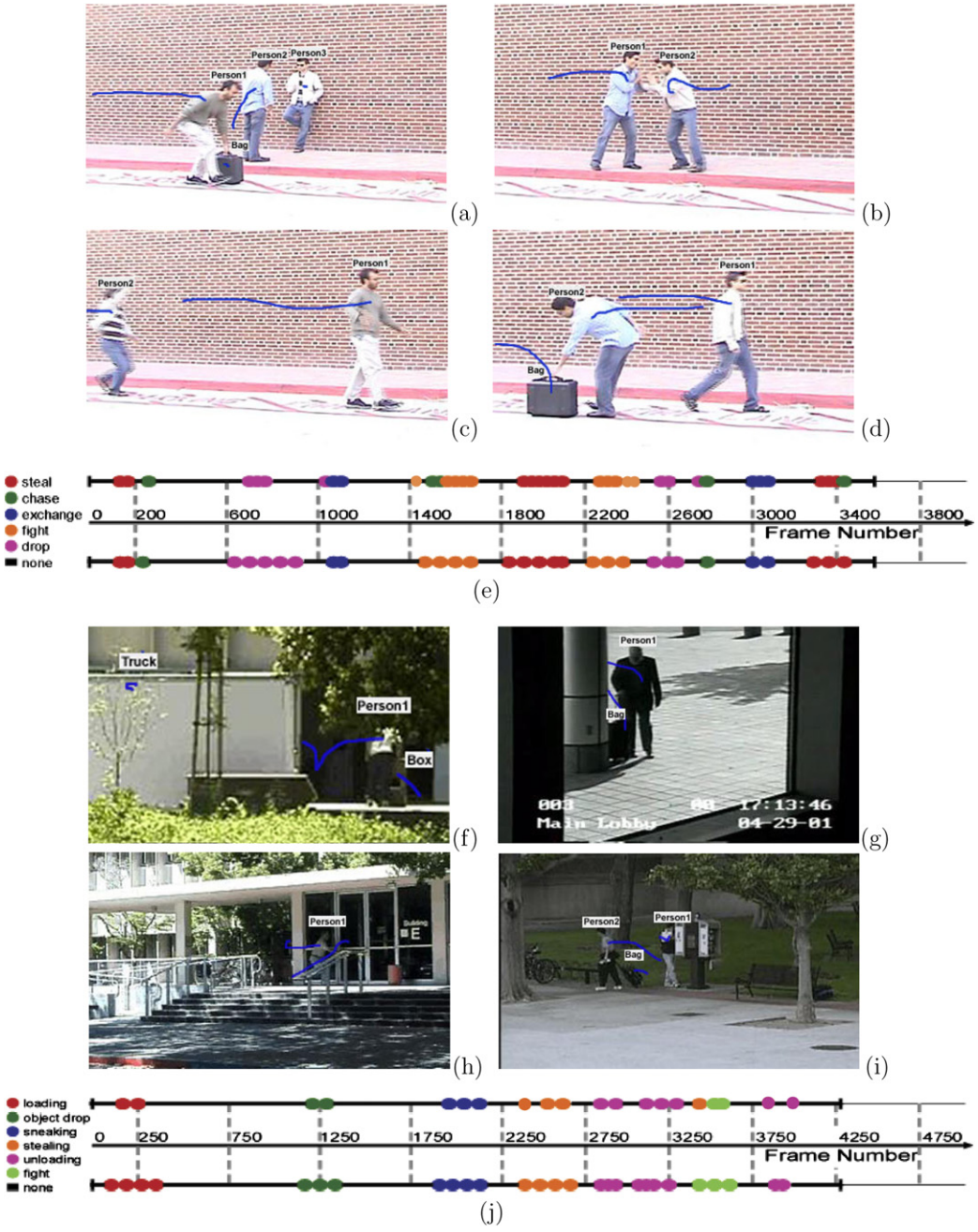


Fig. 7. Event detection results using normalized cuts for surveillance domain test videos. (a)–(d) represent frames 159, 2388, 2874, and 3125 respectively of surveillance video1 consisting of 3580 frames. (e) Time indexed clustering results for surveillance video1 where the top bar shows the actual event detection results, and the bottom bar denotes the ground truth of the events. (f)–(i) represent frames 223, 1084, 2191, and 2703 respectively of surveillance video2 consisting of 4256 frames. (j) Time indexed clustering results for surveillance video2.

jectories, the temporally correlated sub-events were detected in real-time and were further utilized for event learning. A list of all unique sub-events for the surveillance, railroad monitoring and meeting domains are detailed in Table 2.

Using the learnt event models, event detection in novel video proceeded by estimating the weight matrices for each event. Further, normalized cuts algorithm is applied to obtain event clusters in the novel video. The results for event detection using normalized cuts are summarized in Figs. 6, 7, and 8 for the meeting, surveillance and railroad monitoring domains respectively. The precision and recall values for test videos are estimated using

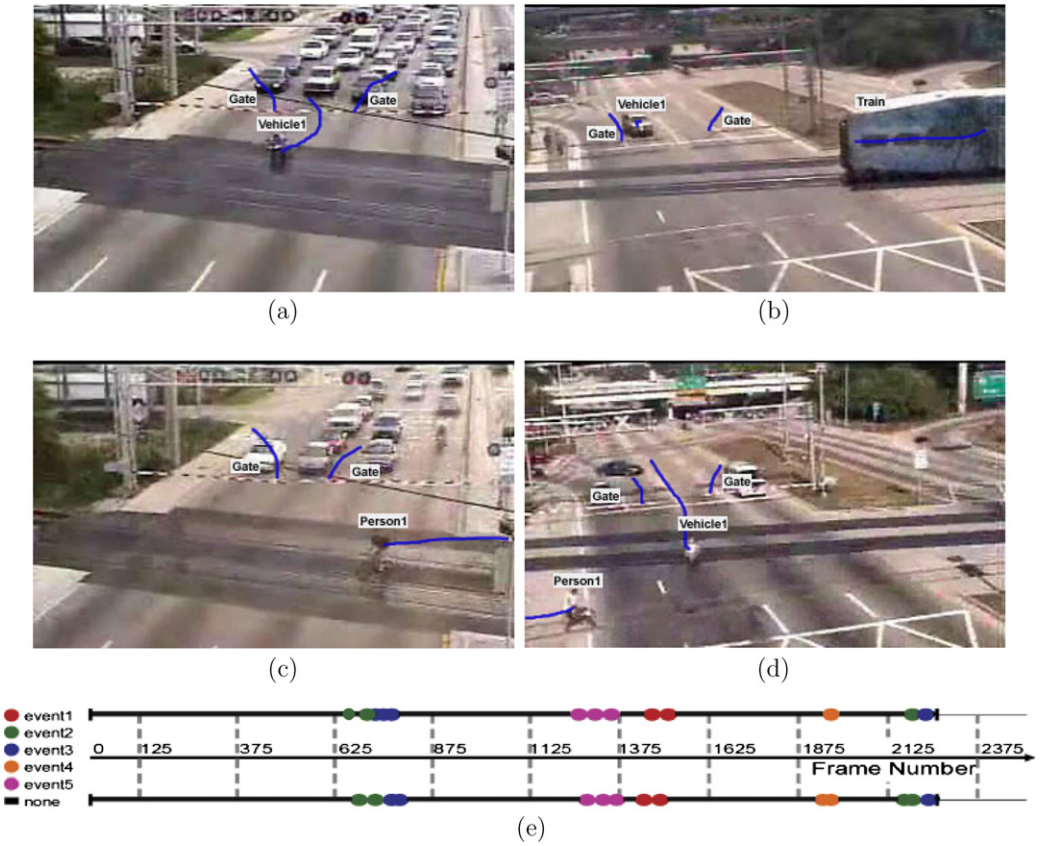


Fig. 8. Event detection results using normalized cuts for railroad monitoring domain test video. (a)–(d) represent frames 740, 1354, 1966, and 2251 respectively of railroad monitoring video consisting of 2260 frames. (e) Time indexed clustering results for railroad monitoring video where the top bar shows the actual event detection results, and the bottom bar denotes the ground truth of the events.

Table 3
Results of testing videos in meeting, surveillance and railroad monitoring domains

Test video	Frames	Events	Sub-events	Precision %	Recall %
Meeting	1551	15	224	92.8	87.9
Surveillance Video1	3580	13	335	92.5	88.9
Surveillance Video2	4256	12	209	77.5	88.5
Railroad monitoring	2260	9	307	85.6	79.8

$$Precision = \frac{\sum_{i,j} \psi(tde_i^j)}{\sum_{i,j} \psi(de_i^j)} \quad \text{and} \quad Recall = \frac{\sum_{i,j} \psi(tde_i^j)}{\sum_{i,j} \psi(te_i^j)}$$

respectively, where $\psi(tde_i^j)$ is the *true detected sub-events*, $\psi(de_i^j)$ is the *detected sub-events*, and $\psi(te_i^j)$ is the *true sub-events* belonging to the ith cluster of the jth event. A summary of event detection results with precision and recall values is described in Table 3.

We also performed automatic annotation of events and sub-events in videos of different domains using our P-CASE representation. Firstly, we automatically generated case-frames in real-time, corresponding to the detected sub-events. Fig. 9 shows snapshots of individuals interacting in an unconstrained environment and their corresponding sub-event representations. Secondly, these sub-events were encoded in the graph of detected sub-events, and events were segmented using *Ncut*. These segmented events were further utilized for automated event annotation of the videos in the meeting, surveillance and railroad monitoring domains. Event-based retrieval of video is an interesting application of this video annotation scheme, which will be explored in our future research.

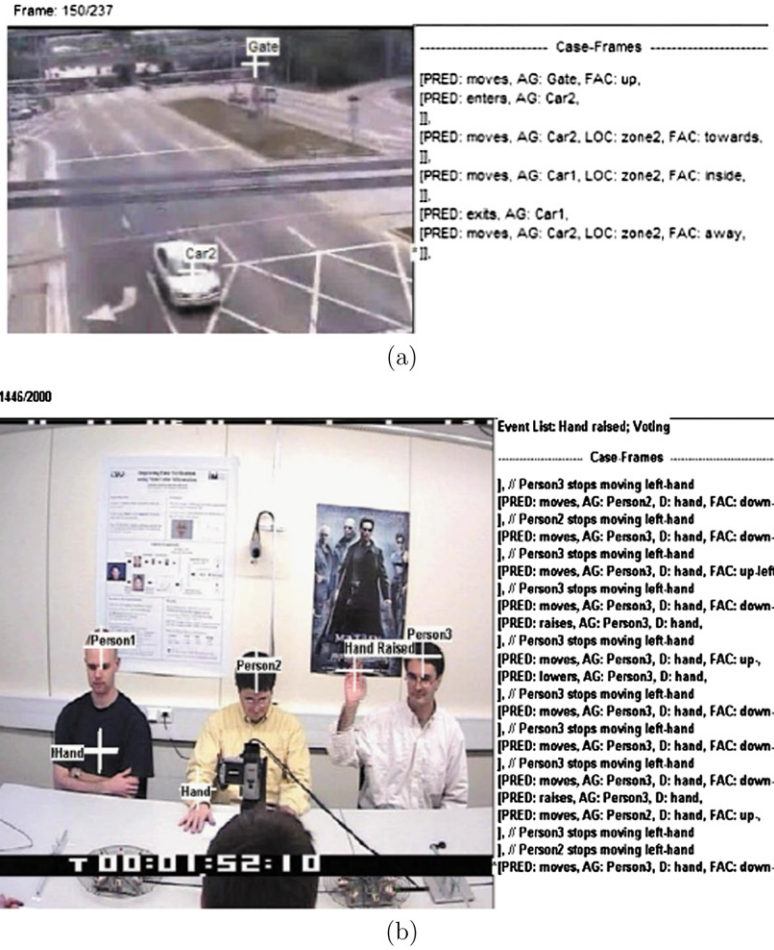


Fig. 9. On-line extended CASE representation of video sequences. (a) Representation at frame 150/237 for the railroad monitoring video (b) PETS sequence representation at frame 1446/2000.

In light of the presented results, we discuss how our method deals with the questions/problems of other methods mentioned earlier. The first problem was developing event models that are scalable to the number of agents involved in the event. It is evident from Figs. 6, 7, and 8 that all the videos involved multiple agent events that are detected with an average precision of 87.1% and an average recall of 86.2%. In order to build event models that are scalable to the number of agents involved in an event, we model the sub-event dependency for all agents simultaneously i.e. which sub-event occurs more frequently after another sub-event for a given event. Thus, our event model is agnostic to the number of agents involved in the event. The second problem was developing event detection methods that deal with independent agent actions. From Fig. 8, it is evident that there are multiple vehicles and objects that are simultaneously moving and their sub-events causes interference in the actual sub-event sequence. By using graph clustering techniques for event detection, we cluster events with high edge weights within the cluster and thus, this method segments out independent agent actions as those actions have low edge weights with the rest of the sub-events belonging to the event.

7. Conclusion

The problem of detecting events in a video involving multiple agents and their interaction was identified. Event models were learnt from training videos that had variations in the number of agents and the temporal order of sub-events. Event learning was formulated in a probabilistic framework, and the learnt event models were used for event detection in novel videos. The main advantages of our event model are: The temporal relations are more descriptive

relationships between sub-events compared to the low level abstract relationship models of HMMs, Dynamic Bayesian Networks etc; The event model does not make any assumptions about the length of an event; The event model is scalable to the number of agents involved in an event since it models the sub-event dependencies instead of the agent processes.

Event detection was treated as a graph theoretic clustering of sub-events having high association within the event clusters and low association outside the clusters. We demonstrated our event detection framework on videos in the railroad monitoring, surveillance and meeting domains. The main advantages of our event detection scheme are: The event detection does not make any assumptions about the length of an event since the graph clustering method clusters highly correlated events of any length; The event detection is scalable to the number of agents involved in an event since the graph clustering method clusters highly correlated events involving any number of agents.

An event representation was developed to cater to the temporal variations in the sub-events, and event ontologies were automatically extracted from training videos. We also proposed a novel event graph representation for the detected events in video sequences, having temporal relationships between sub-events. Hence, unlike almost all previous work, we use both temporal structure and an environment descriptor simultaneously to represent an event. We are interested in several future directions of this work including inference of causality in video sequences and event-based retrieval of video.

Acknowledgements

This material is based upon work funded in part by the US Government. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

Appendix A. Proof of equivalency for W and $\tilde{W}$ based minimizations

Given W, the global criterion for minimization of Ncut function is given by:

$$\begin{aligned} Ncut(A,B) &= \min\left[\frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}\right] \\ &= \min\left[\frac{\sum_{i\in A, j\in B} P(v_j|v_i) + \sum_{i\in A, j\in B} P(v_i|v_j)}{\sum_{i\in A, k\in I} P(v_k|v_i)} + \frac{\sum_{i\in A, j\in B} P(v_j|v_i) + \sum_{i\in A, j\in B} P(v_i|v_j)}{\sum_{j\in B, k\in I} P(v_k|v_j)}\right] \end{aligned}$$

where $I = A \cup B$, and since W is symmetric therefore $P(v_j|v_i) = P(v_i|v_j)$. Thus the above equation is equivalent to:

$$Ncut(A,B) = \min\left[\frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{i\in A, k\in I} P(v_k|v_i)} + \frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{j\in B, k\in I} P(v_k|v_j)}\right]. \tag{A.1}$$

Similarly, given $\tilde{W}$, the global criterion for minimization of Ncut function is given by:

$$\begin{aligned} Ncut(A,B) = \min\Bigg[&\frac{\sum_{i\in A, j\in B} 2P(v_j|v_i) + \sum_{i\in A, j\in B} 2P(v_i|v_j)}{\sum_{i\in A, k\in I} P(v_k|v_i) + \sum_{i\in A, k\in I} P(v_i|v_k)} \\ &+ \frac{\sum_{i\in A, j\in B} 2P(v_j|v_i) + \sum_{i\in A, j\in B} 2P(v_i|v_j)}{\sum_{j\in B, k\in I} P(v_k|v_j) + \sum_{j\in B, k\in I} P(v_j|v_k)}\Bigg], \end{aligned}$$

and since $\tilde{W} = \hat{W} + \hat{W}^T$, where $\hat{W}$ is upper triangle matrix, therefore $P(v_i|v_j) = P(v_i|v_k) = P(v_j|v_k) = 0$. Thus, the above equation is reduced to:

$$Ncut(A,B) = \min\left[\frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{i\in A, k\in I} P(v_k|v_i)} + \frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{j\in B, k\in I} P(v_k|v_j)}\right]. \tag{A.2}$$

Since both Eqs. (A.1) and (A.2) minimize the same function, it is equivalent to deal with W and $\tilde{W}$.

Appendix B. P-CASE representation for railroad monitoring domain

Given the P-CASE representation of an event, we find the most likely sequence of sub-events by calculating the maximum likelihood estimate of all event instances using:

$$E_{ML}(e_i) = \underset{(e_i)}{\operatorname{argmax}}\, P(e_i|E) \tag{B.1}$$

where, e_i are all the event instances belonging to the event E, where $P(e_i|E)$ is computed using:

$$P(e_i|E) = P(s_1, s_2, \ldots, s_n|E) = P(s_1|E)\prod_{j=1}^{n-1} P\left(s_{j+1}, t_j^{j+1}|s_j, E\right)P(s_{j+1}|s_j, E)$$

where, $s_1, \ldots, s_n$ are the sub-events belonging to event instance e_i, n are the number of sub-events in the event instance, and t_j^{j+1} is the temporal relationship between s_j and s_{j+1}. Thus, the likelihood estimate of the following event instance belonging to the object passing P-CASE (shown in Fig. 5) is calculated by:

$$P(\text{holds} \xrightarrow{\text{Overlaps}} \text{moves} \xrightarrow{\text{After}} \text{passes} \xrightarrow{\text{Overlaps}} \text{holds} \xrightarrow{\text{Overlaps}} \text{moves} \mid \text{Object\_Passing})$$

$$= P(\text{holds})\,(P(\text{moves} \mid \text{holds})\,P(\text{moves,OVERLAPS} \mid \text{holds}))\,(P(\text{passes} \mid \text{moves})\,P(\text{passes,AFTER} \mid \text{moves}))\,(P(\text{holds} \mid \text{passes})$$

$$P(\text{holds,OVERLAPS} \mid \text{passes}))\,(P(\text{moves} \mid \text{holds})\,P(\text{moves,OVERLAPS} \mid \text{holds}))$$

$$= 1.0 \times (1.0 \times 0.46) \times (0.15 \times 0.75) \times (1.0 \times 0.86) \times (1.0 \times 1.0) = \mathbf{0.0445}$$

The following is the most likely P-CASE representation for the 10 events in the railroad monitoring domain. Note that for ease of notation and visualization, the temporal relations are given inside the case-frame as was done in CASE^{E} [17,18], instead of being on an edge between case-frames. Thus, we automatically estimate the most likely sequence of sub-events for all the events, and is the same representation that was derived manually by sifting through hours of videos in [17].

Domain entities

Vehicle	A vehicle in the universe
Person	A person in the universe
Train	A train on the tracks
Gate	Gate at the railroad crossing
Signal	Signal at the railroad crossing
Zone1	Zone covering the area of activation for the signal
Zone2	Zone covering a designated high-risk area
Tracks	The tracks that the train travels on

Domain predicates

Moves, *Enters*, *Exits*, *Switches*, *Signals*, *Breaks*, *Collides*, *Stops*.

Domain events

(1) **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle stops outside Zone2**
[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:
[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:
[**PRED**: Stops, **AG**: Vehicle, **LOC**: Zone2, **FAC**: Outside, **AFTER**: Moves]]]]

(2) **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle stops inside Zone2**
[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:
[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:
[**PRED**: Stops, **AG**: Vehicle, **LOC**: Zone2, **FAC**: Inside, **AFTER**: Moves]]]]

(3) **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle breaks the gate arm while entering Zone2**
[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:
[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:
[**PRED**: Enters, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves, **SUB**:
[**PRED**: Breaks, **AG**: Vehicle, **OBJ**: Gate, **DURING**: Enters]]]]]

(4) **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle breaks the gate arm while exiting Zone2**
[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:
[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:
[**PRED**: Exits, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves, **SUB**:
[**PRED**: Breaks, **AG**: Vehicle, **OBJ**: Gate, **DURING**: Exits]]]]]

(5) **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle enters while gate is in motion**
[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:
[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:
[**PRED**: Enters, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves]]]]

(6) **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle exits while gate is in motion**
[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:
[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:
[**PRED**: Exits, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves]]]]

(7) **Vehicle collides with train**
[**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Inside, **SUB**:
[**PRED**: Moves, **AG**: Vehicle, **FAC**: Inside, **LOC**: Zone2, **DURING**: Move, **SUB**:
[**PRED**: Collides, **AG**: { Vehicle, Train }, **AFTER**: Moves]]]

(8) **Person being hit by train**
[**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Inside, **SUB**:
[**PRED**: Moves, **AG**: Person, **FAC**: Inside, **LOC**: Zone2, **DURING**: Move, **SUB**:
[**PRED**: Collides, **AG**: { Person, Train }, **AFTER**: Moves]]]

(9) **Person enters zone2 while signal was switched on**
[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **SUB**:
[**PRED**: Moves, **AG**: Person, **LOC**: Zone2, **FAC**: Towards, **OVERLAPS**: Switches, **SUB**:
[**PRED**: Enters, **AG**: Person, **LOC**: Zone2, **AFTER**: Moves]]]

(10) **Train entering zone2 while gates are in motion**
[**PRED**: Moves, **OBJ**: Gates, **FAC**: Down, **SUB**:
[**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Towards, **OVERLAPS**: Moves, **SUB**:
[**PRED**: Enters, **AG**: Train, **LOC**: Zone2, **DURING**: Moves]]]

References

[1] A. Ali, J.K. Aggarwal, Segmentation and recognition of continuous human activity, in: IEEE Workshop of International Conference on Computer Vision, 2001, pp. 28–38.
[2] J.F. Allen, G. Ferguson, Actions and events in interval temporal logic, Journal of Logic Computation 4 (5) (1994) 531–579.
[3] D. Ayers, M. Shah, Monitoring human behavior from video taken in an office environment, Image and Vision Computing 19 (2001) 833–846.
[4] N. Babaguchi, R. Jain, Event detection from continuous media, in: Proc. of International Conference on Pattern Recognition, 1998, pp. 1209–1212.
[5] N. Badler, Temporal scene analysis: Conceptual description of object movements, University of Toronto Technical Report No. 80, 1975.
[6] J. Ben-Arie, K.R. Rao, Optimal template matching by non-orthogonal image expansion using restoration, International Journal of Machine Vision and Applications 7 (2) (1994) 69–81.
[7] J. Ben-Arie, Z. Wang, P. Pandit, S. Rajaram, Human activity recognition using multidimensional indexing, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (8) (2002) 1091–1104.
[8] M. Brand, Understanding manipulation in video, in: International Conference on Face and Gesture Recognition, 1997, pp. 94–99.
[9] M. Brand, V. Kettnaker, Discovery and segmentation of activities in video, IEEE Transactions of Pattern Analysis and Machine Intelligence 22 (8) (2000) 844–851.
[10] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions of Pattern Analysis and Machine Intelligence 25 (5) (2003) 564–577.
[11] C. Ding, Tutorial on spectral clustering, in: International Conference on Machine Learning, 2004.
[12] C.J. Fillmore, The case for CASE, in: E. Bach, R. Harms (Eds.), Universals in Linguistic Theory, Holt, Rinehart, and Winston, New York, 1968, pp. 1–88.
[13] N. Friedman, K. Murphy, S. Russell, Learning the structure of dynamic probabilistic networks, in: Proc. Conference on Uncertainty in Artificial Intelligence (UAI), Madison, WI, 1998, pp. 139–147.
[14] S. Gong, T. Xiang, Recognition of group activities using dynamic probabilistic networks, in: Proc. of International Conference on Computer Vision, 2003, pp. 742–749.
[15] N. Haering, R.J. Qian, M.I. Sezan, A semantic event-detection approach and its application in wildlife hunts, IEEE Transactions on Pattern Analysis and Machine Intelligence 10 (6) (2000) 857–868.
[16] A. Hakeem, M. Shah, Ontology and taxonomy collaborated framework for meeting classification, in: Proc. of International Conference on Pattern Recognition, 2004, pp. 219–222.
[17] A. Hakeem, Y. Sheikh, M. Shah, CASE^{E}: A hierarchical event representation for the analysis of videos, in: Proc. of American Association of Artificial Intelligence (AAAI), 2004, pp. 263–268.
[18] A. Hakeem, M. Shah, Multi-agent event learning, detection, and representation in videos, in: Proc. of American Association of Artificial Intelligence (AAAI), 2005, in press.
[19] M. Harville, D. Li, Fast, integrated person tracking and activity recognition with plan-view templates from single stereo camera, in: Proc. of Computer Vision and Pattern Recognition, 2004, pp. 398–405.
[20] S. Hongeng, F. Nevatia, R. Bremond, Video-based event recognition: Activity representation and probabilistic recognition methods, Computer Vision and Image Understanding 96 (2) (2004) 129–162.
[21] S. Intille, A.F. Bobick, A framework for recognizing multi-agent action from visual evidence, in: Proc. of American Association of Artificial Intelligence (AAAI), 1999, pp. 518–525.
[22] Y.A. Ivanov, A.F. Bobick, Recognition of visual activities and interactions by stochastic parsing, IEEE Transactions of Pattern Analysis and Machine Intelligence 22 (2000) 852–872.
[23] O. Javed, M. Shah, Tracking and object classification for automated surveillance, in: Proc. of European Conference on Computer Vision, 2002, pp. 343–357.
[24] O. Javed, M. Shah, D. Comaniciu, A probabilistic framework for object recognition in video, in: Proc. of International Conference on Image Processing, 2004.
[25] D. Koller, N. Heinze, H.H. Nagel, Algorithmic characterization of vehicle trajectories from image sequences by motion verbs, in: Proc. of Computer Vision and Pattern Recognition, 1991, pp. 90–95.
[26] F. Lv, R. Nevatia, Recognition and segmentation of 3-D human action using HMM and multi-class AdaBoost, in: European Conf. on Computer Vision, vol. 4, 2006, pp. 359–372.
[27] N. Maillot, M. Thonnat, A. Boucher, Towards ontology based cognitive vision, in: Proc. of International Conference of Vision Systems, 2003, pp. 44–53.
[28] R. Mann, A. Jepson, Towards the computational perception of action, in: Proc. of Computer Vision and Pattern Recognition, 1998, pp. 794–799.
[29] G. Medioni, F. Cohen, I. Brémond, S. Hongeng, R. Nevatia, Event detection and analysis from video streams, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (8) (2001) 873–889.
[30] D. Minnen, I. Essa, T. Starner, Expectation grammars: Leveraging high-level expectations for activity recognition, in: Proc. of Computer Vision and Pattern Recognition, 2003, pp. 626–632.
[31] R. Nevatia, T. Zhao, S. Hongeng, Hierarchical language-based representation of events in video streams, in: IEEE Workshop on Event Mining, Madison, WI, 2003.
[32] R. Nevatia, J. Hobbs, B. Bolles, An ontology for video event representation, in: IEEE Workshop on Event Detection and Recognition, 2004.
[33] N.T. Nguyen, H.H. Bui, S. Venkatesh, G. West, Recognising and monitoring high-level behaviours in complex spatial environments, in: Proc. of Computer Vision and Pattern Recognition, 2003, pp. 620–625.

[34] N. Oliver, B. Rosario, A. Pentland, A Bayesian computer vision system for modelling human interaction, in: Proc. of International Conference of Computer Vision Systems, 1999, pp. 255–272.
[35] B. Ozer, T. Lv, W. Wolf, A bottom-up approach for activity recognition in smart rooms, in: Proc. of International Conference on Multimedia Expo, 2002, pp. 917–920.
[36] V. Parameswaren, R. Chellappa, Human action–recognition using mutual invariants, Computer Vision and Image Understanding 98 (2005) 295–325.
[37] V. Parameswaren, R. Chellappa, Using 2D project invariance for human action recognition, International Journal of Computer Vision 66 (1) (2006).
[38] C. Pinhanez, A. Bobick, Human action detection using PNF propagation of temporal constraints, in: Proc. of Computer Vision and Pattern Recognition, 1998, pp. 898–904.
[39] C. Rao, A. Yilmaz, M. Shah, View-invariant representation and recognition of actions, International Journal of Computer Vision 50 (2002) 203–226.
[40] Y. Rui, P. Anandan, Segmenting visual actions based on spatio-temporal motion patterns, in: Proc. of Computer Vision and Pattern Recognition, 2000, pp. 11–118.
[41] K. Shafique, M. Shah, A noniterative greedy algorithm for multiframe point correspondence, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (1) (2005) 51–65.
[42] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions of Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.
[43] Y. Shi, Y. Huang, D. Minnen, A. Bobick, I. Essa, Propagation networks for recognition of partially ordered sequential action, in: Proc. of Computer Vision and Pattern Recognition, 2004, pp. 862–870.
[44] Y. Shi, A. Bobick, I. Essa, Learning temporal sequence model from partially labeled data, in: Proc. of Computer Vision and Pattern Recognition, 2006.
[45] J.M. Siskind, Visual event classification via force dynamics, in: Proc. of American Association of Artificial Intelligence (AAAI), AAAI Press, Menlo Park, CA, 2000, pp. 149–155.
[46] P. Smith, M. Shah, N. Lobo, Integrating and employing multiple levels of zoom for activity recognition, in: Proc. of Computer Vision and Pattern Recognition, 2004, pp. 928–935.
[47] N. Vaswani, A.R. Chowdhury, R. Chellapa, Activity recognition using the dynamics of the configuration of interacting objects, in: Proc. of Computer Vision and Pattern Recognition, 2003, pp. 633–642.
[48] N. Vaswani, A.R. Chowdhury, R. Chellapa, Shape activity: A continuous-state HMM for moving/deforming shapes with application to abnormal activity detection, IEEE Transactions on Image Processing 14 (2005) 1603–1616.
[49] Z. Wang, J. Ben-Arie, Optimal ramp edge detection using expansion matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (11) (1996) 1092–1098.
[50] C.R. Wren, A. Azarbayejani, T. Darrel, A.P. Pentland, Pfinder: Real-time tracking of the human body, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (7) (1997) 780–785.
[51] G. Xu, Y. Ma, H. Zhang, S. Yang, A HMM based semantic analysis framework for sports game event detection, in: Proc. of International Conference on Image Processing, 2003, pp. 25–28.
[52] L. Zelnik-Manor, M. Irani, Event-based analysis of video, in: Proc. of Computer Vision and Pattern Recognition, 2001, pp. 123–130.
[53] H. Zhong, J. Shi, M. Visontai, Detecting unusual activity in video, in: Proc. of Computer Vision and Pattern Recognition, 2004, pp. 819–826.
[54] CAVIAR: Context Aware Vision using Image-based Active Recognition, More details at http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/.
[55] PETS: Performance Evaluation for Tracking and Surveillance, More details at http://www.cvg.cs.rdg.ac.uk/PETS-ICVS/pets-icvs-db.html.
[56] VACE: Video Analysis and Content Exploitation, More details at http://www.informedia.cs.cmu.edu/arda/vaceI_integrate.html.